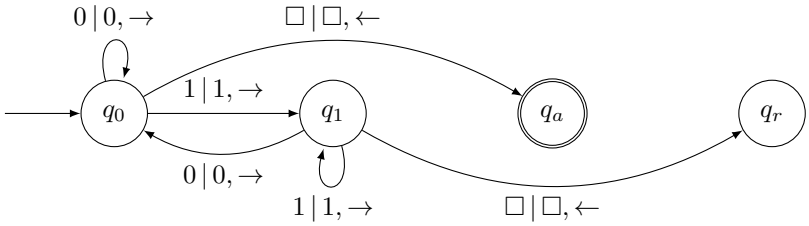
DM5 : Corrigé

1. En n'écrivant pas les symboles blancs sur le ruban se trouvent à gauche du premier symbole non blanc ou à droite du dernier symbole non blanc on obtient la suite de configurations suivantes :

$$(q_0, 001101, 0), (q_0, 001101, 1), (q_0, 001101, 2), (q_1, 001101, 3), (q_1, 001101, 4), (q_r, 001101, 3)$$

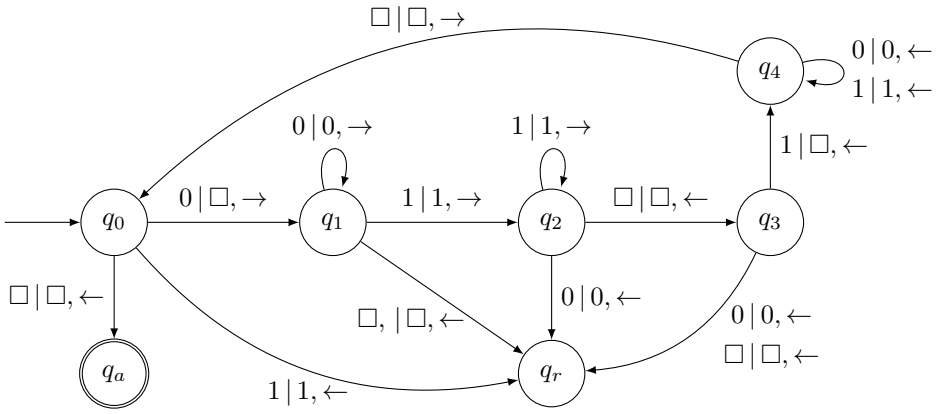
Le mot est donc rejeté. Le langage accepté par cette machine est 0^*1^* : on parcourt de gauche à droite le mot en entrée : tant qu'on lit un 0, on progresse. S'il n'y a que des zéros (y compris aucun), on arrive à la fin du mot (signalé par un symbole blanc) et on accepte ; sinon on lit un premier 1. On accepte le mot si on ne lit que des 1 jusqu'à la fin à partir de cet instant et on refuse sinon.

2. On propose la machine suivante :



Le principe est de faire en sorte d'être dans l'état q_i si et seulement si le dernier bit lu est i. Lorsqu'on arrive à la fin du mot, si on est dans q_0, on accepte et si on est dans q_1, on rejette.

3. On propose la machine suivante :



Le principe de cette machine est le suivant : tant que c'est possible, on remplace le premier 0 et le dernier 1 du mot par □. Si on parvient à faire cette opération jusqu'à ce qu'il n'y ait plus de lettre sur le ruban, le mot est accepté, sinon il est rejeté. Dans le détail :

- L'état q_0 permet de lire la première lettre du mot : si c'est 0 cette lettre est remplacée par □, si c'est 1 c'est que le mot a trop de 1 donc on rejette et si il n'y en a pas, on a réussit à réduire le mot à ε donc on accepte.
- L'état q_1 lit des 0 jusqu'à trouver un 1 : dès que cela arrive, on passe dans l'état q_2 dont le but est de lire les 1 jusqu'à la fin du mot. Trouver la fin du mot dans q_1 ou lire un 0 dans q_2 est signe que le mot n'est pas dans le langage souhaité.
- Lorsque q_2 détecte la fin du mot, on passe dans l'état q_3 dont l'objectif est de rebrousser chemin d'un cran de sorte à transformer le dernier 1 en □. Remarquez que si on arrive dans q_3, il est en fait impossible d'emprunter l'une des transitions allant de q_3 vers q_r car la dernière lettre sur le ruban est forcément un 1.
- L'état q_4 sert à parcourir le ruban en sens inverse jusqu'à revenir au début du mot.

4. Si L est rationnel alors L est reconnu par un automate fini déterministe et complet $A = (\Sigma, Q, q_0, F, \delta)$. On construit alors la machine de Turing $M = (Q \cup \{q_a, q_r\}, \Sigma, \Gamma, \delta_M, q_0, q_a, q_r, \square)$ telle que :

- q_a et q_r sont deux nouveaux états n'appartenant pas à Q.

- $\Gamma = \Sigma \cup \{\square\}$.
- Pour tout $q \in Q$ et tout $a \in \Sigma$, $\delta_M(q,a) = (\delta(q,a), a, \rightarrow)$.
- Pour tout $q \in Q$, $\delta_M(q,\square) = \begin{cases} (q_a, \square, \rightarrow) & \text{si } q \in F \\ (q_r, \square, \rightarrow) & \text{sinon.} \end{cases}$

On montre alors que $L(M) = L(A) = L$ ce qui conclut. Le principe est de lire le mot sur le ruban sans modifier ce dernier et en modifiant l'état de M de la même façon qu'il est modifié par A. Lorsqu'on arrive à la fin du mot, on accepte si l'état atteint est final dans A et on rejette sinon.

5. C'est toujours le même argument : l'ensemble de tous les langages est infini non dénombrable alors que l'ensemble des langages de RE est dénombrable. Ceci est du au fait qu'à toute machine de Turing on peut associer une unique chaîne de caractères binaire et finie décrivant ses états et transitions.

   Il existe donc une infinité non dénombrable de langages qui ne sont pas dans RE.

6. Supposons par l'absurde que L_D soit dans RE. Alors il est accepté par une machine de Turing M et il existe $k \in \mathbb{N}$ tel que $M = M_k$. Comme N est bijective, il existe un mot $u \in \Sigma^\star$ tel que $N(u) = k$. On a alors deux possibilités :

   - Si $u \in L_D$, alors par définition de ce langage, $u \notin L(M_{N(u)}) = L(M_k) = L(M) = L_D$.
   - Si $u \notin L_D$, alors par définition de ce langage, $u \in L(M_{N(u)}) = L_D$.

   On aboutit à une contradiction qui assure que $L_D \notin$ RE et donc en particulier $L_D \notin$ R. S'il existait une machine de Turing M s'arrêtant sur toute entrée et telle que $L(M) = \overline{L_D}$, alors en rendant acceptant son état rejetant et rejetant son état acceptant, on aurait construit une machine qui décide L_D ce qu'on vient de montrer faux. Donc $\overline{L_D}$ est tout aussi indécidable que L_D.

7. Montrons plutôt la contraposée : si $L_A \leq L_B$ et L_B est décidable alors L_A aussi. Il existe une machine M qui termine sur toute entrée u avec $f(u)$ écrit sur son ruban telle que f vérifie $u \in L_A \Leftrightarrow f(u) \in L_B$. Il existe une machine M' qui termine sur toute entrée v en acceptant v si ce mot est dans L_B et en le refusant sinon.

   On construit alors la machine M'' suivante : elle prend une entrée $u \in \Sigma_A^\star$, lance le calcul de M sur u puis lance le calcul de M' sur le mot $f(u)$ à présent sur le ruban et accepte son entrée si et seulement si M' accepte son entrée. Cette machine termine sur toute entrée puisque c'est le cas de M et de M' et accepte son entrée si et seulement si elle appartient à L_A.

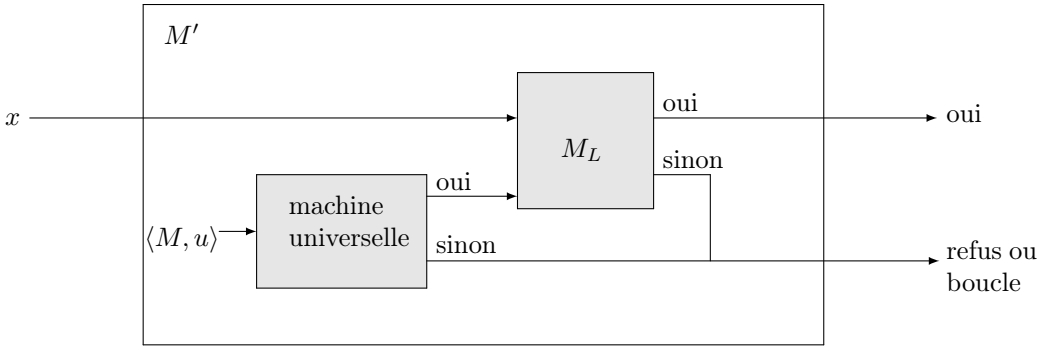
8. Soit $u \in \Sigma^\star$. Pour déterminer $N(u)$, il suffit d'énumérer tous les mots de $\Sigma^\star$ et pour chacun de vérifier s'il est égal à u. Il existe un entier i tel qu'on obtienne cette égalité et alors $N(u) = i$.

   On procède de même pour déterminer $M_{N(u)}$: on énumère tous les encodages de machines de Turing jusqu'à la trouver. Il ne reste plus qu'à concaténer l'encodage de $M_{N(u)}$ avec u, le tout séparé d'un symbole spécial pour calculer $\langle M_{N(u)}, u \rangle$.

9. Soit $u \in \Sigma^\star$. On a :
$$u \in \overline{L_D} \Leftrightarrow u \in L(M_{N(u)}) \Leftrightarrow \langle M_{N(u)}, u \rangle \in L_\in \Leftrightarrow f(u) \in L_\in$$
   Comme f est calculable, ceci montre que $\overline{L_D}$ se réduit à $L_\in$ et comme $\overline{L_D}$ est indécidable, $L_\in$ l'est aussi.

10. a) Si $\emptyset \in \mathcal{P}$, alors il suffit de travailler avec $\mathcal{P}' = \text{RE} \backslash \mathcal{P}$ qui est tout aussi non vide et différent de RE que l'est $\mathcal{P}$ et qui ne contient plus le langage vide. On suppose donc dans la suite que $\emptyset \notin \mathcal{P}$.

    b) Dessinons la machine M' :



Les "sinon" dans cette machine correspondent aux refus et aux boucles. On peut bien la construire à partir de M_L et d'une machine universelle ce qui montre que f est calculable. De plus, par construction, $x \in L(M')$ si et seulement si $u \in L(M)$ et $x \in L(M_L)$. On a donc :

$$L(M') = \begin{cases} L(M_L) = L \in \mathcal{P} & \text{si } u \in L(M) \\ \emptyset \notin \mathcal{P} & \text{sinon, puisque } \emptyset \notin \mathcal{P} \text{ d'après 10a).} \end{cases}$$

On en déduit que $L(M') \in \mathcal{P}$ si et seulement si $u \in L(M)$ et cette dernière assertion se réécrit :

$$f(\langle M, u \rangle) \in L_{\mathcal{P}} \Leftrightarrow \langle M, u \rangle \in L_{\in}$$

c) La question précédente montre que $L_{\in}$ se réduit à $L_{\mathcal{P}}$. Comme $L_{\in}$ est indécidable, la question 7 assure que $L_{\mathcal{P}}$ l'est aussi ce qui montre le théorème de Rice.

Culture générale : L'appelation RE (respectivement R) pour l'ensemble des langages semi-décidables (respectivement décidable) vient de "récursivement énumérable" (respectivement "récursifs"). La raison pour laquelle les langages récursivement énumérables sont ainsi appelés est qu'on peut montrer que $L \in RE$ si et seulement si il existe une machine de Turing appelé énumérateur qui écrit sur son ruban tous les mots de L séparés par un symbole blanc sans jamais déplacer sa tête de lecture vers la gauche.